LAPORAN PRAKTIKUM 3
PEMROGRAMAN LANJUT
RECURSIVE

Oleh

Nama	:	Rizqillah
NIM	:	1957301020
Kelas	:	TI 2C
Dosen Pembimbing	:	Musta'inul Abdi, SST., M.Kom.

PROGRAM STUDI TEKNIK INFORMATIKA

JURUSAN TEKNOLOGI INFORMASI DAN KOMPUTER

TAHUN 2021

LEMBAR PENGESAHAN

No. Praktikum	:	03/PPL/2C/TI/2021
Judul	:	Recursive
Nama	:	Rizqillah
NIM / Kelas	:	1957301020 / TI 2C
Jurusan	:	Teknologi Informasi Dan Komputer
Prodi	:	Teknik Informatika
Tanggal praktikum	:	18 Maret 2021
Tanggal penyerahan	:	21 Maret 2021
Nilai	:	

Buketrata, 21 Maret 2021

Dosen Pembimbing,

Musta'inul Abdi, SST., M.Kom.
NIP. 19911030 20190310 1 5

DAFTAR ISI

BAB 1
PENDAHULUAN

1.1 Tujuan

Modul ini mengenalkan suatu teknik pemrograman yang lebih tinggi. Dalam bagian ini Anda akan mempelajari rekursif dan tipe data abstrak.

Setelah menyelesaikan pelajaran ini, diharapkan Anda dapat:

1. Memahami dan menggunakan rekursif
2. Membuat dan mengelola rekursif dengan bagus
3. Memahami maksud penggunaan rekursif dalam pemrograman

1.2 Dasar Teori

Rekursif adalah teknik pemecahan masalah yang powerful dan dapat digunakan ketika inti dari masalah terjadi berulang kali. Tentu saja, tipe dari masalah ini dapat dipecahkan mengunakan perkataan berulang-ulang (i.e., menggunakan konstruksi looping seperti for, while dan do-while).

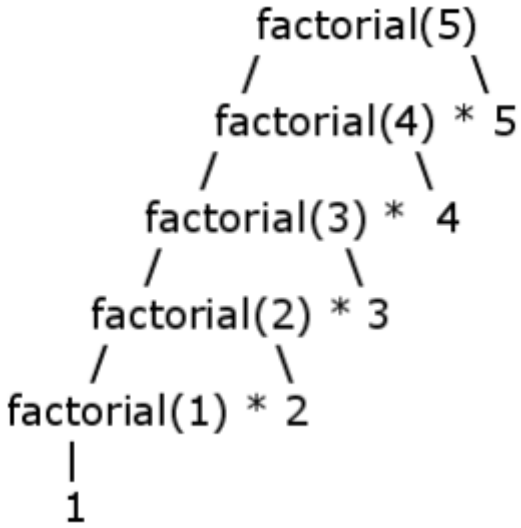
Sesungguhnya, iterasi atau perkataan berulang-ulang merupakan peralatan yang lebih efisien jika dibandingkan dengan recursif tetapi recursion menyediakan solusi yang lebih baik untuk suatu masalah. Pada rekursif, method dapat memanggil dirinya sendiri. Data yang berada dalam method tersebut seperti argument disimpan sementara kedalam stack sampai method pemanggilnya diselesaikan.

1.2.1 Rekursif vs Iterasi

Untuk pengertian yang lebih baik dari rekursif, mari kita lihat pada bagaimana macammacam dari teknik iterasi. Dalam teknik-teknik tersebut dapat juga kita lihat penyelesaian sebuah loop yang lebih baik menggunakan rekursif dari pada iterasi.

Menyelesaikan masalah dengan perulangan menggunakan iterasi secara tegas juga digunakan pada struktur kontrol pengulangan. Sementara itu, untuk rekursif, task diulangi dengan memanggil sebuah method pengulangan. Maksud dari hal tersebut adalah untuk menggambarkan sebuah masalah kedalam lingkup yang lebih kecil dari pengulangan itu sendiri. Pertimbangan suatu perhitungan yang faktorial dalam penentuan bilangan bulat.

Definisi rekursif dari hal tersebut dapat diuraikan sebagai berikut: factorial(n) = factorial(n-1) * n; factorial(1) = 1. Sebagai contohnya, nilai faktorial dari 2 sama dengan fatorial (1)*2, dimana hasilnya adalah 2. Faktorial dari 3 adalah 6, dimana sama dengan faktorial dari (2)*3.



Dengan iterasi, proses diakhiri ketika kondisi loop gagal atau salah. Dalam kasus dari penggunaan rekursif, proses yang berakhir dengan kondisi tertentu disebut permasalahan dasar yang telah tercukupi oleh suatu pembatasan kondisi. Permasalahan yang mendasar merupakan kejadian yang paling kecil dari sebuah masalah. Sebagai contohnya, dapat dilihat pada kondisi rekursif pada faktorial, kasus yang mudah adalah ketika inputnya adalah 1. 1 dalam kasus ini merupakan inti dari masalah.

Penggunaan dari iterasi dan rekursif dapat bersama-sama memandu loops jika hal ini tidak digunakan dengan benar.

Keuntungan iterasi dibandingkan recursion adalah performance yang lebih baik. Hal tersebut lebih cepat untuk recursion sejak terbentuknya sebuah parameter pada sebuah method yang disebabkan oleh suatu CPU time. Bagaimanapun juga, rekursif mendorong practice software engineering yang lebih baik, sebab teknik ini biasanya dihasilkan pada kode yang singkat yang lebih mudah untuk dimengerti dan juga mempromosikan reuseability pada suatu solusi yang telah diterapkan.

Memilih antara iterasi dan rekursif merupakan masalah dari menjaga keseimbangan antara baiknya sebuah performance dan baiknya software engineering.

1.2.2 Facktorial : Contoh

Listing program berikut ini menunjukkan bagaimana menghitung faktorial menggunakan teknik iterasi.

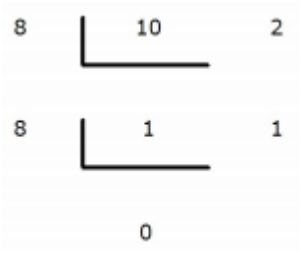
```
class FactorialIter {
    static int factorial(int n) {
      int result = 1;
      for (int i = n; i > 1; i--) {
          result *= i;
      }
      return result;
    }

    public static void main(String args[]) {
      int n = Integer.parseInt(args[0]);
      System.out.println(factorial(n));
    }
}
```

Dibawah ini merupakan listing program yang sama tetapi menggunakan rekursif.

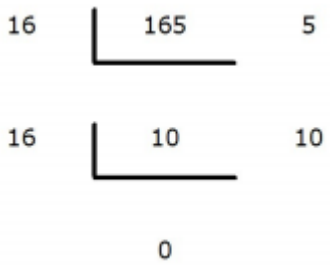
```
class FactorialRecur {
    static int factorial(int n) {
      if (n == 1) { /* The base case */
            return 1;
      }
      /* Recursive definition; Self-invocation */
      return factorial(n-1)*n;
    }

    public static void main(String args[]) {
      int n = Integer.parseInt(args[0]);
      System.out.println(factorial(n));
    }
}
```

1.2.3 Print n in Any Base : Contoh yang lain

Sekarang, pertimbangan dari masalah dalam pencetakkan suatu angka desimal yang nilai basenya telah ditetapkan oleh pengguna. Ingat bahwa solusi dalam hal ini untuk menggunakan repetitive division dan untuk menulis sisa perhitungannya. Proses akan berakhir ketika sisa hasil pembagian kurang dari base yang ditetapkan. Dapat diasumsikan jika nilai input desimal adalah 10 dan kita akan mengkonversinya menjadi base 8. Inilah solusinya dengan perhitungan menggunakan pensil dan kertas.



Dari solusi diatas, 10 adalah sama dengan 12 base 8.
Contoh berikutnya. Nilai input desimalnya adalah 165 dan akan dikonversi ke base 16.



165 adalah sama dengan A5 base 16. Catatan: A=10.

Berikut ini merupakan solusi iterative untuk masalah diatas.

```
class DecToOthers {
      public static void main(String args[]) {
            int num = Integer.parseInt(args[0]);
            int base = Integer.parseInt(args[1]);
            printBase(num, base);
      }

      static void printBase(int num, int base) {
            int rem = 1;
            String digits = "0123456789abcdef";
            String result = "";
            /* the iterative step */
```

```
            while (num!=0) {
                  rem = num%base;
                  num = num/base;
                  result = result.concat(digits.charAt(rem)+"");
            }
            /* printing the reverse of the result */
            for(int i = result.length()-1; i >= 0; i--) {
                  System.out.print(result.charAt(i));
            }
      }
}
```

Berikut ini merupakan recursion untuk masalah yang sama dengan solusi sebelumnya.

```
class DecToOthersRecur {
      static void printBase(int num, int base) {
            String digits = "0123456789abcdef";
            /* Recursive step*/
            if (num >= base) {
                  printBase(num/base, base);
            }
            /* Base case: num < base */
            System.out.print(digits.charAt(num%base));
      }

      public static void main(String args[]) {
            int num = Integer.parseInt(args[0]);
            int base = Integer.parseInt(args[1]);
            printBase(num, base);
      }
}
```

BAB 2
PEMBAHASAN

2.1 Factorial

Dalam matematika, Faktorial dari bilangan bulat positif dari n yang dilambangkan dengan n!, adalah sumber dari semua bilangan bulat positif yang kurang dari atau sama dengan n :

$$n! = n \times (n-1) \times (n-2) \times (n-3) \times \cdots \times 3 \times 2 \times 1\,.$$

Sebagai contoh,

$$5! = 5 \times 4 \times 3 \times 2 \times 1 = 120\,.$$

Nilai 0! adalah 1, menurut konvensi untuk produk kosong.

Operasi faktorial digunakan sebagai bidang matematika, terutama di kombinatorik, aljabar, dan analisis matematika. Penggunaannya yang paling dasar menghitung kemungkinan urutan dan permutasi dari *n* yang berada di objekk yang berbeda.

Faktorial pada fungsi juga dapat berupa nilai ke argumen non-bilangan bulat sambil mempertahankan properti terpentingnya dengan cara mendefinisikan $x! = \Gamma(x + 1)$, dimana Γ adalah fungsi gamma; ini tidak ditentukan saat *x* adalah bilangan bulat negatif.

2.1.1 Factorial Dengan Iterasi

Program :

```
public class FaktorialIter {
    static int factorial(int n) {
        int result = 1;
        for (int i = n; i > 1; i--) {
            result *= i;
        }
        return result;
    }

    public static void main(String args[]) {
        int n = 5;
        System.out.println(factorial(n));
    }
}
```

Hasil :

```
120
BUILD SUCCESSFUL (total time: 1 second)
```

Analisa :

```
public class FaktorialIter {
    static int factorial(int n) {
```

➔ **Membuat Public class FaktorialIter, beserta Method bertipe data int dengan nama factorial menerima parameter int n.**

```
        int result = 1;
```

➔ **Deklarasi nilai variabel result = 1**

```
        for (int i = n; i > 1; i--) {
```

➔ **Perulangan for dengan nilai i = n, dan mengecek kondisi apakah i > 1.**

```
            result *= i;
```

➔ **Jika true, maka melakukan operasi result = result * i.**

```
        }
        return result;
```

➔ **Jika kondisi for menghasilkan false, maka mengembalikan nilai result.**

```
    }

    public static void main(String args[]) {
```

➔ **Membuat method main, yang nantinya akan dibaca pertama oleh program ketika berjalan.**

```
        int n = 5;
```

➔ **Deklarasi nilai n bertipe data int dengan value 5.**

```
        System.out.println(factorial(n));
```

➔ **Mencetak kelayar hasil dari pemanggilan method factorial dengan memasukkan nilai parameter n(5) yang telah dideklarasi sebelumnya.**

```
    }
}
```

2.1.2 Factorial Dengan Recursif

Program :

```
public class FaktorialRecur {
    static int factorial(int n) {
        if (n == 0) {
            /* The base case */
            return 1;
        } else {
            /* Recursive definition; Self-invocation */
            return factorial(n - 1) * n;
        }
    }

    public static void main(String args[]) {
//        int n = Integer.parseInt(args[0]);
        int n = 5;
        System.out.println(factorial(n));
    }
}
```

Hasil :

```
120
BUILD SUCCESSFUL (total time: 0 seconds)
```

Analisa :

```
public class FaktorialRecur {
    static int factorial(int n) {
```

➔ **Membuat public class FaktorialRecur. Dan membuat method factorial yang bertipe data int dengan menerima parameter variabel n bertipe data int.**

```
        if (n == 0) {
            return 1;
```

➔ **Melakukan perkondisian if, jika nilai n == 0, maka true, dan akan mengembalikan nilai 1 ke yang memanggil method.**

```
        } else {
            return factorial(n - 1) * n;
```

➔ **Jika false, maka melakukan pemanggilan method factorial dengan diisi nilai n-1. Dan akan melakukan hal tersebut terus-terusan sampai nilai n menjadi sama dengan 0. Dan ketika nilai n mencapai 0, maka akan mereturn 1 ke pemanggil, dan pemanggil dikala itu bernilai n=2, maka akan dihitung 1*2, dan hasilnya dikembalikan ke pemanggil ketika**

```
      n bernilai 3, dan mereturn 2*3 ke pemanggil sebelumnya.
      Yaitu ketika nilai n=4, dan akan dihitung 4*6, hasil
      perhitungan dikembalikan ke pemanggil sebelumnya, yaitu
      dimana nilai n=5. Dan akan dihitung 5*24. Dan hasil
      perkalian tersebut kemudian akan dikembalikan si pemanggil
      yang ada di method main.
        }
    }

    public static void main(String args[]) {
        int n = 5;
        System.out.println(factorial(n));
   ➔ Mencetak hasil dari pemanggilan method factorial yang diisi
      dengan parameter variabel n(5).
    }
}
```

2.2 Decimal To Other

Desimal (Basis 10)

Sistem bilangan yang menggunakan basis 10 disebut Desimal. Kata desimal berasal dari akar kata Latin decem (sepuluh). Bilangan desimal terdiri 10 angka D={0,1,2,3,4,5,6,7,8,dan 9}.

Contoh : Bilangan 456(desimal)
Pada bilangan tersebut, digit 3 berarti 4 ratusan, 5 berarti 5 puluhan, dan 6 berarti 6 satuan. Sehingga, 4 mempunyai arti paling besar di antara tiga digit yang ada. Digit ini bertindak sebagai digit paling besar (*Most Significant Digit, MSD*).

Biner (Basis 2)

Sistem bilangan yang menggunakan basis 2 disebut Biner. Kata biner berasal dari akar kata Latin bine (double). Bilangan biner terdiri 2 angka B={0 dan 1}.

Bilangan biner disebut binary digit atau bit. 4 bit dinamakan nibble dan 8 bit dinamakan byte atau oktet. Sejumlah bit yang dapat diproses komputer untuk mewakili suatu karakter (dapat berupa huruf, angka atau lambang khusus) dinamakan word. Sebuah komputer dapat memproses data satu word yang terdiri dari 4 sampai 64 bit. Sebagai contoh, sebuah komputer yang menggunakan mikroprosesor 32 bit dapat menerima, memproses, menyimpan dan mengirim data atau instruksi dalam format 32 bit.

Contoh: Bilangan 1010(biner)
Bit paling kiri (dari depan anda) ini bertindak sebagai digit paling besar (*Most Significant Bit, MSB*). Sedangkan bit paling kanan (dari depan anda) bit paling kecil (*Least Significant Bit, LSB*).

$1010 = (1 \text{ X } 2^3) + (0 \text{ X } 2^2) + (1 \text{ X } 2^1) + (0 \text{ X } 2^0) = 8+0+2+0$
Maka, 1010(Biner) = 10(Desimal)

Octal (Basis 8)
Sistem bilangan yang menggunakan basis 8 disebut Oktal. Kata oktal berasal dari akar kata Latin octo (delapan). Bilangan Oktal terdiri dari 8 angka O={0,1,2,3,4,5,6,dan 7}.

Contoh: Bilangan 56(oktal)
Penyelesaiannya sebagai berikut:
$56 = (5 \text{ X } 8^1) + (6 \text{ X } 8^0) = 40+6$
Maka, 56(Octal) = 46(Desimal)
HexaDesimal (Basis 16)
Sistem bilangan yang menggunakan basis 16 disebut HexaDesimal. Kata hexadesimal berasal dari akar kata yunani hex (enam) dan Latin decem (sepuluh). Bilangan Hexadesimal Terdiri dari 16 angka H={0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,dan F}.

Bilangan A,B,C,D,E,F
Pada sistem hexa desimal, A=10, B=11, C=12, D=13, E=14 dan F=15.

Contoh: Bilangan 1A(hexa)
Penyelesaiannya sebagai berikut:
$1A = (1 \text{ X } 16^1) + (10 \text{ X } 16^0) = 16+10$
Maka, 1A(Hexa) = 26(Desimal)

2.2.1 Decimal To Other Iterasi

Program :

```
public class DecToOthers {
    public static void main(String args[]) {
        int num = 2;
        int base = 2;
        printBase(num, base);
    }

    static void printBase(int num, int base) {
        int rem = 1;
        String digits = "0123456789abcdef";
        String result = "";
        /* the iterative step */
        while (num != 0) {
            rem = num % base;
            num = num / base;
            result = result.concat(digits.charAt(rem) + "");
        }

        /* printing the reverse of the result */
        for (int i = result.length() - 1; i >= 0; i--) {
            System.out.print(result.charAt(i));
        }
    }

}
```

Hasil :

```
10BUILD SUCCESSFUL (total time: 0 seconds)
```

Analisa :

```
public class DecToOthers {
    public static void main(String args[]) {
        int num = 2;
        int base = 2;
```

➔ **Deklarasi nilai variabel num dan base bertipe data int, dengan nilai 2.**

```
        printBase(num, base);
```

➔ **Memanggil method printBase dengan memasukkan nilai parameter num dan base.**

```
    }

    static void printBase(int num, int base) {
```

➔ **Membuat method printBase bertipe voi yang menerima parameter num dan base yang bertipe data int.**

```
        int rem = 1;
        String digits = "0123456789abcdef";
        String result = "";
```

➔ **Membuat variabel rem bertipe data int dengan nilai 1. Variabel digits dan result bertipe String, nilai digits = "0123456789abcdef", dan result = "".**

```
        while (num != 0) {
```

➔ **Melakukan perulangan while selama nilai num tidak sama dengan 0.**

```
            rem = num % base;
```

➔ **Menghitung nilai num %(modulus) base, hasilnya disimpan pada variabel rem.**

```
            num = num / base;
```

➔ **Menghitung nilai num / base, hasil disimpan di variabel num.**

```
            result = result.concat(digits.charAt(rem) + "");
```

➔ **Melakukan penggabungan huruf yang diambil pada index dengan nilai rem, dan diambil menggunakan charAt dengan parameter index rem. Dan menggabungkan dengan spasi, disini menggunakan concat agar String yang disimpan bisa disimpan berkali-kali tanpa menghilangkan nilai String didalamnya. Dan String yang disimpan selanjutnya akan berada didepannya. Oleh karenanya, String yang disimpan bisa membentuk lebih dari 1 huruf yang diambil.**

```
        }

        for (int i = result.length() - 1; i >= 0; i--) {
```

➔ **Melakukan perulangan for, dengan nilai integer i = panjang result-1. Dan mengecek kondisi apakah i >= 0.**

```
            System.out.print(result.charAt(i));
```

➔ **Jika true, maka akan mencetak tiap-tiap kata yang ada pada variabel result, dengan menggunakan charAt dengan parameter dari nilai i.**

```
        }
    }
}
```

2.2.2 Decimal To Other Recursif

Program :

```
public class DecToOtherRecur {
    static void printBase(int num, int base) {
        String digits = "0123456789abcdef";
        /* Recursive step*/
        if (num >= base) {
            printBase(num / base, base);
        }
        /* Base case: num < base */
        System.out.print(digits.charAt(num % base));
    }

    public static void main(String args[]) {
        int num = 2;
        int base = 2;
        printBase(num, base);
    }
}
```

Hasil :

```
10BUILD SUCCESSFUL (total time: 0 seconds)
```

Analisa :

```
public class DecToOtherRecur {
    static void printBase(int num, int base) {
```

➔ **Membuat method printBase bertipe void yang menerima parameter num dan base yang bertipe data integer.**

```
        String digits = "0123456789abcdef";
```

➔ **Deklarasi variabel digith dengan tipe data String dan value “0123456789abcdef”.**

```
        if (num >= base) {
            printBase(num / base, base);
        }
```

➔ **Mengecek kondisi if, apakah nilai num >= base. Jika true, maka melakukan pemanggilan method printBase dengan nilai parameter 1 = num/base, dan parameter 2 = base.**

```
        System.out.print(digits.charAt(num % base));
```

➔ **Mencetak nilai variabel digits yang terletak pada index = num & base ke layar.**

```
    }
```

```
    public static void main(String args[]) {
        int num = 2;
        int base = 2;
```
➔ **Deklarasi variabel num dan base dengan tipe data integer, dan dengan value 2.**

```
        printBase(num, base);
```
➔ **Memanggil method printBase dengan mengisi nilai parameter dengan variabel num dan base.**

```
    }
}
```

2.3 Palindrom

Palindrom adalah kata, rangkaian kata, atau bilangan yang terbaca sama, baik dari depan maupun dari belakang, seperti *kodok, radar,* dan *taat.* Palindrom adalah sebuah kata, frasa, angka maupun susunan lainnya yang dapat dibaca dengan sama baik dari depan maupun belakang (spasi antara huruf-huruf biasanya diperbolehkan). Kata "palindrom" berasal dari bahasa Yunani.

Menurut buku *Mother Tongue: English & How It Got That Way* (hal. 227): "Palindrom … berumur setidaknya 2.000 tahun". Palindrom Latin "Sator Arepo Tenet Opera Rotas" sangat unik karena ia akan megulang kalimatnya lagi jika kita membentuk kata dari huruf pertama setiap kata kemudian disambung dengan huruf kedua setiap kata, dan seterusnya. Karena itu ia juga dapat disusun dalam sebuah kotak yang dapat dibaca secara vertikal maupun horisontal:

```
S A T O R
A R E P O
T E N E T
O P E R A
R O T A S
```

Palindrom ada dalam banyak bahasa-bahasa Barat, terutamanya di bahasa Inggris. Meskipun begitu, gelar "bahasa palindrom" jatuh pada bahasa Finlandia. Selain itu, palindrom juga ada dalam bahasa-bahasa non-Barat, contohnya bahasa Jepang, bahasa Tionghoa dan bahasa Korea.

Dalam Bahasa Indonesia kalimat seperti di bawah ini juga merupakan palindrom:

- Aku suka rajawali, bapak. Apabila wajar, aku suka (oleh Benjamin Goodspeed Zimmer).
- Kasur ini rusak.
- Kasur Nababan rusak.
- Kasur Koh Ahok rusak.

- Ibu Ratna antar ubi.
- Dia haid.
- Ria Irawan nawari air. (bahasa Indonesia gaul/tidak baku)
- Harum semar kayak rames murah. (bahasa Indonesia gaul/tidak baku)
- Tasupi dan Tika sama-sama sakit nadi pusat.
- Ira hamil lima hari.

Kata-kata yang termasuk palindrom misalnya:

- ada, apa, ara, asa, bab, ini, katak, kodok, makam, malam, radar, taat, tamat, dll

Program Palindrome :

```
9  import java.util.Scanner;
10 public class PalindromeRekursif {
11     public static void main(String[] args) {
12         String hasilS;
13         Scanner input = new Scanner(System.in);
14         System.out.print("Input kata yang akan di cek : ");
15         String kata = input.nextLine();
16         String kata2 = kata;
17         kata = kata
18                 .toLowerCase().replaceAll("\\s+", "").replaceAll("-", "");
19         //memanggil fungsi cekPalindrome dan memberi parameter variabel kata
20         boolean hasil = cekPalindrome(kata);
21 
22         if (hasil) {
23             hasilS = "Betul!";
24         } else {
25             hasilS = "Salah!";
26         }
27         System.out.println("Apakah \"" + kata2 + "\" adalah Palindrome? : " + hasilS);
28     }
30     private static boolean cekPalindrome(String kata) {
31         //Cek apakah kata yang dimasukkan cuma 1 huruf
32         if (kata.length() == 1) {
33             return true;
34         }
35         //Mengisi nilai awal dengan huruf pertama pada variabel kata
36         String awal = kata.substring(0, 1);
37         //Mengisi nilai akhir dengan huruf akhir pada variabel kata
38         String akhir = kata.substring(kata.length() - 1, kata.length());
39 
40         //Melakukan pengecekan apakah nilai awal dan akhir tidak sama
41         if (!awal.equals(akhir)) {
42             return false;
43             //jika kata awal dan akhir sama, maka akan memanggil fungsi rekursif
44         } else {
45             return cekPalindrome(kata.substring(1, kata.length() - 1));
46         }
47     }
48 }
```

Hasil :

```
Input kata yang akan di cek : Tasupi dan Tika sama-sama sakit nadi pusat
Apakah "Tasupi dan Tika sama-sama sakit nadi pusat" adalah Palindrome? : Betul!
BUILD SUCCESSFUL (total time: 15 seconds)
```

Analisa :

```
import java.util.Scanner;
```

➔ **Import library Scanner, agar nantinya bisa memasukkan data dari keyboard.**

```
public class PalindromeRekursif {
    public static void main(String[] args) {
```

➔ **Membuat public class PalindromeRekursif, kemudian membuat method main, yaitu suatu method yang akan dibaca pertama ketika program berjalan.**

```
        String hasilS;
        Scanner input = new Scanner(System.in);
```

➔ **Membuat variabel hasilS dengan tipe data String, dan objek dari Scanner.**

```
        System.out.print("Input kata yang akan di cek : ");
        String kata = input.nextLine();
```

➔ **Mencetak kalimat ke layar, beserta meminta user menginput nilai yang nantinya disimpan di variabel kata yang bertipe String.**

```
        String kata2 = kata;
```

➔ **Membuat variabel kata2 dengan tipe data String yang diisi nilai dari variabel kata.**

```
        kata = kata.toLowerCase().replaceAll("\\s+",
"").replaceAll("-", "");
```

➔ **Melakukan perubahan nilai pada variabel kata, yaitu mengubahnya menjadi LowerCase, dan membuat seluruh spasi, tab atau lainnya yang berbentuk ruang kosong akan ditiadakan menjadi null. Dan meniadakan semua penulisan tanda kurang(-) menjadi null.**

```
        boolean hasil = cekPalindrome(kata);
```

➔ **Membuat variabel hasil dengan tipe data boolean, dan mengisi data dengan hasil pengembalian dari method cekPalindrome yang diisi parameter variabel kata.**

```
        if (hasil) {
            hasilS = "Betul!";
```

➔ **Mengecek apakah nilai variabel hasil = true. Jika true, maka akan mengisi nilai "Betul!" pada variabel hasilS.**

```
        } else {
            hasilS = "Salah!";
        }
```

➔ **Jika nilai hasil = false. Maka mengisi nilai hasilS menjadi "Salah!".**

```
        System.out.println("Apakah \"" + kata2 + "\" adalah Palindrome? : " + hasilS);
```

➔ **Mencetak kalimat ke layar, beserta nilai dari variabel kata2 dan nilai dari variabel hasilS.**

```
    }

    private static boolean cekPalindrome(String kata) {
```

➔ **Membuat method cekPalindrome dengan tipe boolean, dan menerima parameter variabel kata bertipe data String.**

```
        if (kata.length() == 1) {
            return true;
        }
```

➔ **Mengecek apakah panjang dari variabel kata == 1. Jika true, maka akan langsung me-return true.**

```
        String awal = kata.substring(0, 1);
```

➔ **Mengisi nilai variabel awal yang bertipe data String dengan huruf yang ada di index 0 ke 1, contohnya jika nilai kata = "kodok", maka nilai variabel awal yang pertama adalah "k".**

```
        String akhir = kata.substring(kata.length() - 1, kata.length());
```

➔ **Mengisi nilai variabel akhir bertipe data String dengan cara mengambil huruf terakhir menggunakan metode panjang huruf – 1, maka akan didapatkan nilai index terakhir dari huruf. Misal "kodok", maka panjang huruf – 1 = 4, dan index ke-4 adalah huruf k, dan batasnya sampai ke index ke-5, jadinya huruf yang diisi hanyalah huruf terakhir, yaitu "k".**

```
        if (!awal.equals(akhir)) {
            return false;
```

➔ **Mengecek apakah nilai variabel awal tidak sama dengan nilai akhir. Jika iya, maka akan langsung return true.**

```
        } else {
            return  cekPalindrome(kata.substring(1,  kata.length()
- 1));
        }
```

➔ **Jika false, maka akan memanggil method cekPalindrome dengan mengisi nilai variabel dari nilai variabel kata yang ada di index 1 ke panjang kata – 1(index terakhir) dari huruf.**

```
    }
}
```

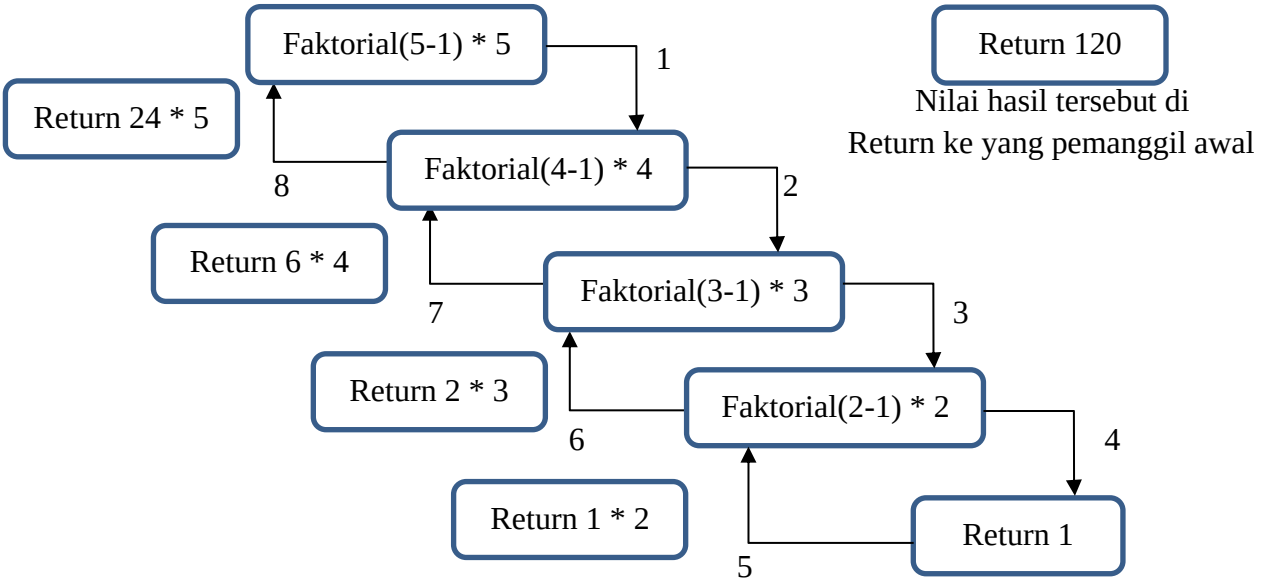
BAB 3
PENUTUP

3.1 Kesimpulan

Dalam pembuatan program yang bersifat rekursif, maka program ketika berjalan pada suatu method, program tersebut akan memanggil methodnya sendiri hingga hasil yang diinginkan tercapai. Dan ketika hasil yang diinginkan tercapai, maka program tersebut akan memberi hasilnya kepada si pemanggil yang sebelum dan sebelumnya, hingga seluruh pemanggilan telah mendapat hasil dari method tersebut. Jika hasil telah ditemukan, maka fungsi rekursif ini akan berhenti.

Metode rekursif ini mirip seperti metode stack, yaitu LIFO(Last In First Out). Dikarenakan nilai yang duluan dimasukkan akan mendapati hasilnya yang paling terakhir, dan akibat fungsi ini berjalan secara memanggil methodnya sendiri. Maka akan melakukan operasi terus-menerus sampai menemukan nilai dari hasil yang diinginkan. Dan setelah menemukan nilai yang diinginkan, program akan berhenti dan hasil yang didapati akan dikirim ke si pemanggil method tersebut.

Adapun contoh diagram berjalannya program factorial yang menggunakan sifat rekursif adalah sebagai berikut :



DAFTAR PUSTAKA

[1] Thamura, Frans. *Modul JENI-intro3: Bab03 Teknik Pemrograman Lanjut*. JENI.

[2] Faktorial. https://id.wikipedia.org/wiki/Faktorial/. Diakses pada 19 Maret 2021.

[3] Pengertian Sistem Bilangan. 09 Juni 2018. https://www.mariobd.com/. Diakses pada 19 Maret 2021

[4] Rudy Setiawan. 4 Sistem Bilangan Komputer. https://www.rsetiawan.com/. Diakses pada 19 Maret 2021

[5] Palindrom. https://kbbi.web.id/palindrom/. Kamus Besar Bahasa Indonesia(KBBI). Diakses pada 19 Maret 2021.

[6] Palindrom. https://id.wikipedia.org/wiki/Palindrom/. Diakses pada 19 Maret 2021.

[7] Berita Hari Ini. Apa Itu Palindrom. https://kumparan.com/ Kumparan. Diakses pada 19 Maret 2021.